Netherlands
organization for
applied scientific
research

TNO-report

TNO Institute for Perception

P.O. Box 23
3769 ZG Soesterberg
Kampweg 5
3769 DE Soesterberg, The Netherlands

Fax +31 3463 5 39 77
Phone +31 3463 5 62 11

IZF 1991 B-2

P.J.M.D. Essens
C.A. McCann*
M.A. Hartevelt

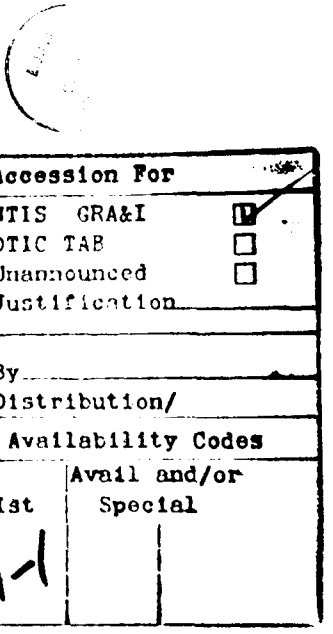
AN EXPLORATORY STUDY OF THE INTERPRETATION OF LOGICAL OPERATORS IN DATABASE QUERYING

*On scientific exchange from Defence and Civil Institute of Environmental Medicine, P.O. Box 2000, North York, Ontario, Canada M3M 3B9

Number of pages: 29

TNO

CONTENTS



Report No.:	IZF 1991 B-2
Title:	An exploratory study of the interpretation of logical operators in database querying
Authors:	Drs. P.J.M.D. Essens, C.A. McCann and drs. M.A. Hartevelt
Institute:	TNO Institute for Perception TNO Defence Research Group: Cognitive Psychology
Date:	March 1991
DO Assignment No.:	B91-34
No. in Program of Work:	733.1

SUMMARY

The use of logical operators in query languages is considered to be a major source of user problems in database querying. The present study investigated whether people untrained in logic could successfully interpret logical operators; and, how errors and latencies are related to the structure of the query. In an experiment, the logical complexity of an SQL-style query formulation was varied in using AND, OR, and NOT operators in either single or combined form. The latency and error data converged to show that subjects had increasing difficulty with queries constructed with a combination of different operators. The inclusion of brackets had a strong positive effect on task performance. Verbal protocols were used to identify sources of errors in query processing. A model of query processing was formulated and predictions latencies and errors on the basis of processing components were tested.

Rap.nr. IZF 1991 B-2

Instituut voor Zintuigfysiologie TNO,
Soesterberg

Een verkennende studie van de interpretatie van logische operatoren bij het bevragen van een database

P.J.M.D. Essens, C.A. McCann en M.A. Hartevelt

SAMENVATTING

Het gebruik van logische operatoren in vraagtalen wordt beschouwd als een belangrijke bron van gebruikersproblemen bij het bevragen van een database. In de huidige studie is onderzocht of mensen die ongetraind waren in logica, met succes logische operatoren konden interpreteren en hoe fouten en responstijden gerelateerd zijn aan de structuur van de formele vraag. In een experiment werd de logische complexiteit gevarieerd van een formele vraag (van het SQL-type) door AND, OR en NOT operatoren te gebruiken enkelvoudig of gecombineerd. De resultaten van de responstijden en fouten gaven beide hetzelfde beeld: de combinatie van verschillende operatoren leverde het grootste probleem op. Het toevoegen van haakjes in de vraagformulering had een sterk positief effect op de taakuitvoering. Verbale protocollen werden gebruikt om bronnen van fouten te identificeren in het verwerken van de vraag. Een model van de verwerking van formele vragen werd geformuleerd en voorspellingen van responstijden en fouten op basis van verwerkingscomponenten werden getest.

1 INTRODUCTION

Database management systems (DBMSs) are now widely used for structuring and storing textual information in applications ranging from small business administration systems to large library cataloguing systems. The user of a DBMS retrieves information from the database by specifying the subset desired along dimensions that are recognized by the database. This involves two major steps. The first is the formulation of a mental representation of the information subset that is desired, based on the user's conception of the way in which the database is structured. In the second step, that representation is transformed into an explicit query using a database query language or environment. The subset defined by the query is then extracted by the DBMS and presented to the user who checks to see if it is the desired set. If it is not, the loop of query creation, submission, extraction and checking is repeated until the user is satisfied.

A variety of interfaces and querying languages have been developed to provide access to data sets (Vassiliou & Jarke, 1984). Many are based on the structure of the language SQL (Chamberlin et al., 1976). Yet studies of such interfaces have demonstrated that people, especially non-programmers, have difficulty using them (Jarke & Vassiliou, 1985). To date, the fundamental research investigating database querying has focussed mainly on the following areas and issues: the user's mental model of the information system; influence of the structure of the information set; the syntax of the query language; and the use of Boolean operators for subset specification, the topic of this study. The remainder of this introduction gives a brief description of the first three issues mentioned, since work on these topics is also relevant to the current study; this is followed by a consideration of research done to date on the use of logical operators in querying.

Users are normally taught procedures for using an information retrieval system, for example, by formal training or from a manual. However, after this training, they may still have a poor mental model of the underlying capabilities, structure, and internal relationships in the information system. This lack of appropriate mental model prevents users from making inferences and predictions about the system's behaviour and inhibits further learning. In a study addressing this issue, Borgman (1986) demonstrated experimentally that subjects trained on simple retrieval tasks via a "conceptual" model of an on-line library catalogue were better able to perform complex tasks that required extrapolation from the basic concepts than those subjects who were trained only on retrieval procedures.

The data contained within an information system are conceptually organized by the designers according to a "data model". There are three generic frameworks in which data is typically structured: the hierarchical, network and relational frameworks (corresponding to the different types of database management systems). The particular subdivision, grouping and linkage of a specific set of data within the selected generic framework forms the data model. Problems can

arise if the structure of the data model is unknown to the user or is inappropriate for the task. Early work by Broadbent and Broadbent (1978) investigated the structuring of classes of objects and the allocation of descriptor terms to the classes. It showed that a person's retrieval of information is much better when the queries are based on the terms that he or she had assigned. Later studies (Lochovsky & Tsichritzis, 1977; Brosey & Shneiderman, 1978) compared the use of different generic data models on query writing.

Retrieval problems are sometimes due to factors at a lower level of human-computer communication, for example, the particular syntax or lexicon used in the query language itself. These issues have been identified in language usability studies carried out during the design of new query environments. Such studies have identified minor, but repeatedly-made errors like omitted punctuation, misspelled terms and the incorrect use of synonyms (Reisner, 1977; Welty, 1985; Thomas, 1975).

In most database interfaces, the user specifies the subset of information that is desired by using Boolean connectives to form a logical combination of attributes that describe the set. Problems can arise when the user's concept of subset formation does not match the Boolean-based one demanded by the computer. The current work is focussed on this particular issue in information retrieval.

There is much incidental and anecdotal comment on the difficulty people have with logical (Boolean) operators like AND and OR in database querying (e.g., (Thompson & Croft, 1989; Cooper, 1988). Marchionini (1989) notes that the Boolean operators provided for filtering the retrievals from an electronic database like an encyclopaedia are important tools for effective use of these information systems. However, in a study of the use of such databases by students, he found that they did not seem to take full advantage of search tools involving Boolean connectives. The AND operator was used as a connective in only one third of the electronic searches; the OR and NOT operators were never used. In the domain of library retrieval, Borgman (1986) found that 25% of subjects learning an SQL-like query language could not pass benchmark tests for system proficiency, although these tests were representative of the searches that were supported. The problem seemed to lie in the use of Boolean logic: more than one quarter of the subjects could not complete simple search tasks involving the use of one index and at most one Boolean operator.

One aspect of the difficulty stems from an incompatibility between natural English usage of the connectives "and" and "or" and their use in database retrieval. Ogden and Kaplan (1986) investigated this problem in detail, showing that the English word "or" is most frequently used to indicate union, but that "and" is often used ambiguously to indicate both union and intersection. Thus the statement "Show the students in grades 10 and 11" implies the union (logical conjunction) operation (the set of students in grade 10 plus the set of students in grade 11), despite the use of the word "and". Since people often attempt to

"translate" the English-language statement of a problem into the query language in a phrase-by-phrase way, the different meanings of the connectives are not taken into account (Reisner, Boyce & Chamberlin, 1975). Ogden and Kaplan propose the incorporation of simple syntactic elements in the retrieval language to permit users to clarify ambiguous logical combinations.

The other difficulty in subset extraction has to do with the actual understanding and use of the logical operators in subset specification. This notion has not been as thoroughly investigated. Wason and Johnson-Laird (1972) found that subjects could more easily describe conjunctive concepts (involving AND) than disjunctive ones (involving OR) and that concepts which involved both conjunctive and disjunctive relations were the hardest to describe. Furthermore, subjects seemed to prefer positive descriptions of concepts rather than negative ones, even though the negative description was more efficient. The results suggested that perhaps the logically naive person does not naturally think in a logically pure manner.

The results of these studies of logical reasoning seem to be borne out in a more recent study by Greene et al (Greene, Devlin, Cannata & Gomez, 1990) in which performance on subset specification using an SQL type query language with explicit Boolean operators is compared with a tabular interface. In one condition, subjects were required to generate simple queries involving AND, OR, NEG, and AND+OR. In another condition they had to choose the SQL query corresponding to a certain English statement. The average number of correct queries generated using SQL was only around 55%, except for the AND queries which were less error prone (73% correct). The time required to generate the query increased in the order AND, OR, NEGATION, AND+OR. Fewer errors were made in the choose condition, but the pattern of time required for selection was the same.

A study by Katzeff (1986) achieved higher success rates for generation of the same kinds of queries using a language essentially equivalent to the Boolean expression part of SQL. Subjects were first trained to interpret query specifications of the following logical forms:

P, NOT P, P AND Q, P OR Q, NOT (P AND Q), NOT (P OR Q)

Query interpretation required subjects to determine whether a given query would retrieve the information presented on an accompanying Venn diagram. They were then tested on query formulation for these same forms of queries and additional new forms:

P AND NOT Q, P OR NOT Q, NOT P AND Q, NOT P OR Q

With feedback and as much time as needed, subjects were able to generate queries in the first set with a success rate of 96%. Furthermore in almost half of the cases in the second set (45%) they were successful in identifying and using a single statement to express the logical restriction, although they had not been trained on them.

In a third investigation, Michard (1982) compared a Venn diagram aid for subset specification with the classical explicit Boolean operators as in SQL. The query problems involved three or four clauses, and various combinations of AND, OR and MINUS. The SQL specification gave rise to four times as many errors in set specification (38%) as the Venn specification method, including errors such as missing brackets and criterion, and incorrect operators.

Although none of the latter three studies focussed specifically on the issue of logical operators in querying, they provide some evidence to confirm the less rigourously derived observations of Marchionini and Borgman, that, indeed, users are often not successful in correctly employing logical operators in SQL querying. However, the results are somewhat contradictory, with the Green et al experiment giving rather higher error rates for SQL query formulation than that by Katzeff. There are several important methodological factors that prevent direct comparison of the results, and, in addition, preclude the drawing of conclusions about precisely where in the querying process the problem with the logical operators actually lies.

All experiments involved the generation of SQL-style queries from test questions posed in natural language. However the exact task of the subject varied: in the Michard study, subjects were required to create the whole query; in the others, only the part involving the logic of subset specification. Even with liberal error scoring methods, as used by Green et al, it is possible that many of the errors in query formulation were due to factors additional to the understanding of logical operators per se, for example, the subjects' memory for details of syntax. It should be noted that the Green et al experiment circumvented this complication by having a separate task condition that required only the recognition of the correct query from a set.

All studies investigated the use of the operators AND and OR, although there were differences in the logical combinations used and in the overall "complexity" (number of operators and structure) of the queries demanded. Two of the experiments tested the NOT operator, whereas the material used by Michard included the MINUS operation instead. The precise logical form of queries used in the experiment by Green et al was not described. The particular combination of logical operators has a large influence on the processing of the query. This factor was deliberately manipulated in the current study.

A further complication in query formulation concerns the influence of the natural language stimulus material on query formulation, especially given the potential confusion between natural language usage of logical connectives and their use in databases as elucidated by Ogden and Kaplan. In addition, there is the more general problem of determining whether the subject actually knows which information subset is requested by the stimulus question. The Green et al study handled this issue nicely by including a confirmatory test phase to check on subjects' understanding of the test question.

An additional difference in task across studies concerned feedback on response. Subjects were given feedback about the correctness of the query in the Katzeff study, and encouraged to try again on queries incorrectly specified. This may explain the better performance in this study compared to that by Green et al or by Michard.

Training for subjects was by the "standard" SQL manual with examples and exercises in the Green et al experiment; by the use of written instructions and Venn diagram test examples in the Katzeff study; and by demonstration of examples in the Michard study. The form and extent of the training may well have a strong influence on subjects' comprehension of the meaning and use of logical operators. Special attention was paid to training and feedback in the current experiment.

In each previous study, subjects had no experience in computer use, although it is possible that they may have had training in logic, which would have influenced ability to do logical manipulations. The level of general education was higher in the Katzeff and Michard studies (university/college versus highschool).

In every case, the experimental part of the study was relatively short, involving the generation of only between 8 and 24 queries. Thus it is not possible to determine how performance using logical operators might change due to more extensive use of the language. Our study tested performance over a long series of trials.

In summary, the studies to date have not specifically addressed the cognitive interpretation of SQL logical operators under well-controlled conditions, in particular the relationship between errors and the specific operators and logical structure of a query. The current study, therefore, is aimed at providing more insight into what, exactly, is the source of difficulty in the use of Boolean operators in subset specification in querying. More precisely, we wanted to see

a Whether people untrained in logic could successfully interpret the logical operators (i.e., intersection, union, negation) as used to specify subsets in standard database querying;
b What kind of errors they made and how the errors related to the structure and complexity of the subset specification.

2 METHOD

There were several considerations that influenced the design of the experiment. First, as this was an exploratory investigation, we wished to keep the subject's task as simple as possible. We wanted to avoid, for example, the complications introduced by use of "and" and "or" in natural language. Furthermore, we wanted to concentrate on the subject's conception of Boolean operators per se and to

avoid the influence of complex processes like language transformation assumed in query writing (Reisner, 1977). We therefore chose query interpretation rather than query writing as the focus of attention, the rationale being that the latter involves the former. In addition, the dataset attributes were limited to four in number to control for the learning of the data model. Second, since it's use is now so widespread, we decided to employ an SQL-style of presentation as the subset specification "language". Third, we choose "naive" subjects - those without training in programming or logic. Furthermore, we limited the amount of training provided to subjects to the minimum necessary to get them going on the task, in the expectation that this strategy would better reveal the errors that they made.
The general task of the subject was to decide whether the SQL-style query presented on the screen would or would not result in the selection of the data base item that was described along with the query. Performance was measured in terms of latencies to evaluate each query and the errors made. Verbal protocol sessions gave qualitative data concerning the query evaluation process.

2.1 Subjects

Twelve students between the ages of 18 and 28 years participated in the experiment. Four of them were university students; the other eight were finishing or had just finished high school. They were selected on the basis of absence of experience with computer programming or any other substantial knowledge of logic. The subjects were paid for their participation.

2.2 Stimuli

The stimulus set was constructed in two steps: first, a set of generic queries differing in the combination of logical operators was created; second, (arbitrary) information about different musical recordings ("LP's") was connected to these queries. The LP-instance consisted of five characteristics or attributes: title of the LP, name of the artist, genre of music, year of recording and rating of the LP. The query was a SQL-type subset specification with two or three conditions ("clauses") referring to LP-instance. The clauses, which in some cases were negated by the logical operator NOT, were connected by logical operators AND or OR. For each trial, a query and an instance of a LP were presented. The subjects had to decide whether the LP-instance did or did not match the query. A sample stimulus is shown in Fig. 1. In the example the LP-instance does not match the query: although the artist is not Will Bazar, the year is not greater than 1970; furthermore, the genre does not match. The title of the LP was never used in the query. A notation employed henceforth in the paper to describe queries uses A, B, and C to represent the clauses in the query; thus the sample query can be written NOT A AND B OR C.

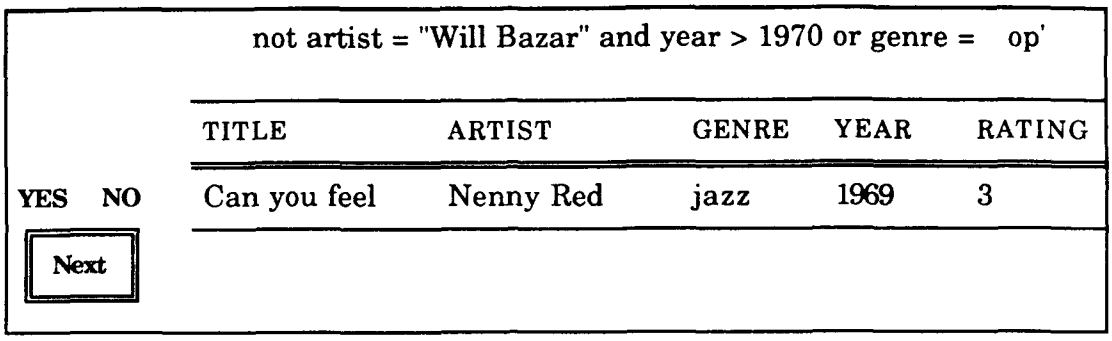


Fig. 1 A sample stimulus showing the query (top line), the LP-instance (bottom line), response buttons, and a button to call the next stimulus.

The combinations of logical operators used in this experiment are shown in Table I. There were four main query categories consisting of two subcategories with combinations of logical operators. Each main query category was crossed with three levels of NOT (none, one or two). The position of the NOT(s) was varied across the A, B, or C clauses. The "mixed" case appeared with and without parentheses. Parentheses were used only in combination with the second logical operator, i.e., NOT A OR (NOT B AND C). No default operator precedence rules were used, the rationale being that this would add an additional degree of complexity. Thus AND and OR were treated as equal in priority. Brackets were used to indicate grouping of the clauses.

Table I Schematic representation of the 24 combinations of logical operators used in the experiment.

	Query categories			
	Single	Double	Mixed	Mixed()
	Logical operator combinations			
	AND;OR	AND AND;OR OR	AND OR;OR AND	AND (OR);OR (AND)
0-NOT				
1-NOT				
2-NOT				

Two main sets of 34 stimuli representing the range of combinations of logical operators and having different query content were formed. The within-clause comparators (=,< >,<,>) were evenly distributed over these queries. Unintended peculiarities of the LP-instance in the query part of the stimulus were

controlled by varying the position and/or content of the LP-instance between the two subsets of stimuli. Queries in which the logical operators differed only in the position of the NOT were distributed over the two subsets in order to reduce the number of stimuli. The two subsets were replicated four times for a total of eight blocks. The content of the LP-instance part of the stimulus was varied between blocks so that the number of "yes" and "no" responses was evenly distributed.
A third set of 40 stimuli (for the verbal protocol sessions) representing most combinations of logical operators, and having the same constraints as the two main sets was also formed. The full set of logical combinations in this set is shown in the Appendix.

2.3 Procedure

The task of the subjects was to evaluate whether a certain LP described in the LP-instance would be selected by a given query. Latency and error data were collected in a so-called "speed session", in which subjects were instructed to perform the task as quickly as possible. The speed session consisted of the eight blocks of 34 (randomly ordered) stimuli. Between the blocks was a short pause. Before and after the speed session there was a "verbal protocol session" in which the subject was asked to think aloud during the evaluation of the queries. The two blocks of the verbal protocol sessions were completely identical and consisted of 40 (randomly ordered) stimuli each.

At the start of a session, subjects were given written instructions about the experiment and the task. The task was explained primarily by examples. No reference to truth tables or the explicit truth or falseness of the clauses in the examples was made. No specific reference to precedence of operators was made. Subjects were told that brackets meant a grouping of the clauses. A series of sample queries were presented, for instance:

artist = 'Will Bazar' and year > 1970 or genre = 'pop'.

The explanation accompanying this query was "The selected LP's have to be made by Will Bazar after 1970 or be pop LP's." A list of six LP-instances was presented with the query and the result of every comparison of query to instance was given for the subject to study and confirm. The effect of the AND and OR were summarised in the following manner: "When an AND is used, then the LP has to have both characteristics to be selected. When an OR is used, then the LP must have one (or all) characteristics." To test whether subjects had learned the logical operators adequately twelve test queries (six on paper and six on the screen) were given after the instructions. The subjects were required to speak out aloud the steps that were taken in the evaluation of these queries. The experimenter confirmed correct evaluations and made short comments on errors ("look again", "are you sure?"). During the experiment no feedback was given about errors.

The stimuli were presented in black on a white 19-inch high-resolution computer screen. The experiment was self-paced; the subject called the next stimulus by clicking on the NEXT-button (see Fig. 1). Between the stimuli was a one-second time gap. A warning signal indicated the arrival of next stimulus. Subjects responded by clicking with the mouse on one of the two fields labeled YES and NO. They were allowed to correct their response. All responses were recorded automatically in the computer. In the verbal protocol sessions subjects were asked to think aloud when processing a stimulus. The verbal protocols were recorded on tape.

3 RESULTS

The latency between presentation of the stimulus and the first response in the speed session was used in further analysis. This response was also analysed for errors.

3.1 Latency data

The distribution of the latency data was slightly skewed and showed a long tail with some extreme large values. On the basis of this distribution, latency values greater than 25 s (n=27 out of 3264) were eliminated from the latency dataset. Furthermore, all cases of incorrect response (the errors) were removed. This reduced the dataset from 3264 to 3070 points. The mean latency to process queries was 8.7 with a standard deviation (sd.) of 3.5 s.

Analyses of variance were performed on the data for the different logical operators and their combinations. The two subsets of stimuli used in the speed session were grouped together, because no differences were found. Mean latencies for the four query categories, single, double, mixed, mixed(), crossed with the three levels of NOT are presented in Table II. The difference between the single and double category is a reflection of the extra processing time of a 3-clause versus a 2-clause query. No significant difference was found in latencies to process single and double queries containing AND operator(s) versus those containing OR (means 8.2 s and 8.1 s, respectively). The differences between double and mixed, and double and mixed() were not significant. The latencies of queries with parentheses ("mixed()") showed a significant improvement compared to those without them ("mixed"), $[F(1,1443)=34.8,\ p<.001]$. An analysis of the position of AND-OR in the queries of the mixed categories indicated no order effect.

The effect of number of NOTs was significant $[F(2,3064)=57.6,\ p<.001]$. A post-hoc test showed that the effect was due to the difference in latencies between 0-NOT and 1-NOT $[F(1,3064)=58.8,\ p<.001]$. No significant higher order interactions were found for the query categories and the NOT's.

Table II Average response latencies (s) for the four query and three NOT categories.

	Query categories				
	Single	Double	Mixed	Mixed()	mean(sd)
0-NOT	6.2	8.3	8.9	8.3	7.7 (3.2)
1-NOT	7.0	9.4	10.3	8.9	9.1 (3.7)
2-NOT	8.1	9.9	10.7	9.1	9.5 (3.9)
mean(sd)	6.9 (3.0)	9.1 (3.3)	10.1 (3.9)	8.8 (3.5)	8.7 (3.5)

Visual scanning of the data showed similar performance of the subjects across different logical combinations, although latencies differed from subject to subject. In particular the university students processed the queries faster than the high school students. For the two groups of students an analysis of variance across the four query categories and NOT categories indicated a significant effect of groups [$(F(1,3062) = 127.6,\ p < .001$]. No higher order interactions of these factors were found. In a two way ANOVA a significant effect of blocks and a significant higher order interaction with student groups was found, respectively, $F(7,3054) = 13.5,\ p < .001$; $F(7,3054) = 3.6,\ p < .001$. Latencies dropped over the 8 speed blocks from 10 to 6.6 s for the university students and from 9.9 to 8.9 s for the high school students. The performance of both groups started off the same, but that of the university students improved more than that of the high school students.

3.2 Error data

The first main analyses considered the differences between query categories, split out in separate analyses (Pearson chi square) of single vs. double, double vs. mixed, mixed vs. mixed(). Mean error rates for the four query categories, single, double, mixed, mixed(), and the levels of the NOT are presented in Table III. The difference between the single and double category is not significant. There is no significant difference in error rates between AND and OR with two and three clause queries pooled together. The difference between double and mixed was significant ($\chi^2(1) = 12.2,\ p < .001$). The queries with parentheses ("mixed()") showed significantly fewer errors compared to those without ("mixed"),[$\chi^2(1) = 5.2,\ p < .025$]. An analysis of the order of AND-OR in the queries of the mixed categories indicated no order effect.

Table III Errors (%) for the four query and three NOT categories as percentages of the number of observations (n).

	Query categories				
	Single	Double	Mixed	Mixed()	Mean(n)
0-NOT	1.6 (384)	2.4 (336)	7.9 (240)	4.2 (192)	3.6 (1152)
1-NOT	7.3 (192)	4.9 (288)	7.3 (288)	3.8 (288)	5.7 (1056)
2-NOT	4.7 (192)	4.9 (288)	8.3 (288)	6.6 (288)	6.2 (1056)
Mean(n)	3.8 (768)	3.9 (912)	7.8 (816)	4.9 (768)	5.1 (3264)

The NOT effect is significant [$\chi^2(1)=9.2$, $p<.005$]. No significant higher order interactions were found for the query categories and the NOT's.

For the two groups of students a χ^2 across the four query categories and NOT categories indicated no effect of groups. No higher order interactions were found. There was a significant effect for blocks [$\chi^2(7)=17.7$ $p<.025$]. The error rate dropped from 4.9% to 1.7% in the last block. However, no significant higher order interaction with student groups was found.

The results of the latency and error data can be summarised as follows (Fig. 2):

(i) largest latencies and highest error rates are found in the "mixed" category; smallest latencies and lowest error rates are found in the "single" category

(ii) latencies increase significantly in the "double" category compared to the "single", but error rates do not increase

(iii) error rates increase significantly in the "mixed" category" compared to the "double", but the higher latencies are not significant

(iv) latencies and error rates decrease significantly in the "mixed()" category compared to the "mixed".

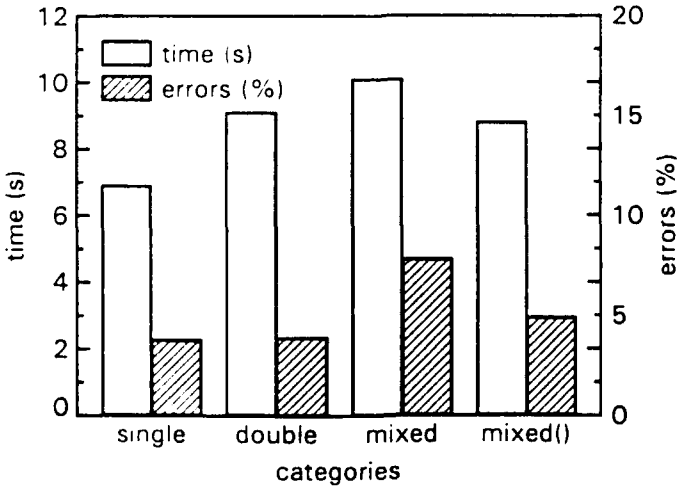


Fig. 2 Latency (s) and error (%) data for the four query categories.

3.3 Verbal Protocols

An analysis of the verbal protocol sessions indicated that subjects read the query aloud, processing it from left to right and evaluating clause by clause. Sometimes the whole query was read first; at other times clauses were read and evaluated immediately. The instance information against which the clauses were evaluated was seldom read aloud.

An analysis of the error data from the verbal protocol (VP) blocks (Table IV) showed that errors were quite high in the four query categories during the first VP block, especially for the complex queries (AND/OR). Average error rate for the first block was 12.3% (n=480), but it reached as high as 22.9% for complex queries with two NOTs. The overall rate dropped to 4.4% in the second verbal protocol block.

Table IV Error data (%) from the verbal protocols of the first and second block for the four query and the three NOT categories as percentages of the number of observations in the categories (n).

	Query categories														
	Single			Double			Mixed			Mixed()			Mean(n)		
	Block														
	1	2	(n)	1	2	(n)	1	2	(n)	1	2	(n)	1	2	(n)
0-NOT	4.2	0.0	(24)	8.3	0.0	(24)	20.8	8.3	(24)	8.3	0.0	(24)	10.4	2.1	(96)
1-NOT	0.0	4.2	(48)	5.6	4.2	(72)	12.5	4.2	(48)	12.5	4.2	(48)	7.4	4.2	(216)
2-NOT	4.2	0.0	(24)	22.9	6.3	(48)	20.8	8.3	(48)	22.9	6.3	(48)	19.6	5.9	(168)
Mean	2.1	2.1	(96)	11.8	4.2	(144)	17.5	6.7	(120)	15.8	4.2	(120)	12.3	4.4	(480)

3.3.1 **Processing errors**

To permit a more detailed analysis, the protocols were coded in terms of processing errors and were examined for specific user strategies. Two levels of query processing had been identified in a previous pilot study: the processing of the clauses and the processing of the logical operators using the results of the clauses as input. The following seven categories of processing errors, distributed over the two levels, were observed. They ranged from misreading clauses to incorrectly interpreting the logical structure of the query.

Mistakes concerning the value (truth or falseness) of a clause can be caused by errors in reading the clause, or errors in evaluating the clause (i.e., comparing the clause with the related instance information, sometimes in combination with a NOT). The following four categories contained these cases.

1 *Reading a clause incorrectly.* Comparators (=, <, >, < >) were often misread. For example, subjects read "smaller than", instead of "greater than"; or "is equal to" instead of "is unequal to". Also, numerals (e.g. 1963) were misread, especially the numerical values that were used in clauses with "year" and "rating". These misreadings did not always result in an incorrect assessment of the clause. Only those misreadings that did were counted.
2 *Missing a NOT while reading.* A NOT operator standing before a clause was sometimes missed in the subject's reading of the clause and was therefore not incorporated in further evaluation, resulting in an incorrect assessment of the value of the clause.
3 *Incorrect evaluation of a clause.* The clause had to be compared to the information given in the LP-instance. Even though a clause was correctly read, subjects could come to an incorrect conclusion about the truth or falseness of the clause after this comparison.
4 *Incorrect evaluation of a NOT clause.* Wrongly evaluated clauses that were combined with a NOT operator were scored separately in this group.

On the query level, the processing of the logical operators and the combining of the results of the clause assessment are the key activities. The verbal protocols enabled us to keep track of the locus of attention of the subjects while they were processing a query. Errors found at this level are the incorrect omission of a clause and incorrect conclusions about the value of the query.
5 *Incorrectly omitting a clause.* Sometimes subjects skipped the evaluation of a relevant clause; this was scored as an omission.
6 *Incorrect evaluation of the query.* In some cases queries whose clauses had been correctly read and evaluated were still processed incorrectly, resulting in an incorrect response. This category contains those errors that could not be assigned to any other.
7 *Redundant evaluation of a clause.* Most of the time subjects correctly bypassed evaluation of the clauses that were redundant in a query. However, sometimes they did process a redundant clause. Processing of these clauses did not change the logical sense of the query. For example, if the first clause of the query A OR B OR C is found to be TRUE, evaluation of the two other clauses is redundant. This category is not associated with errors in response, and so should be considered separately from the preceding six.

The verbal protocols from the first and last sessions of the experiment were coded using these error categories. For each subject, two analyses were performed: an analysis of the incorrect trials in terms of the processing errors (error categories 1-6); and an analysis of the processing of queries with redundant clauses in the query (error category 7). As a general rule, errors observed in processing were assigned to a category only if the error led to an incorrect evaluation of the query against the instance (except for the redundant clauses, category 7). Table V gives the distribution of errors relative to the total number of trials-with-errors for the two verbal protocol blocks. Some incorrect trials contained more than one error count (in the same or different categories). If

there was more than one error in the same category in a query evaluation, this was scored as a single error. Thus, the percentages do not sum to a 100. In block one there were 59 (12.3%) incorrectly assessed queries, and in block two there were 21 (4.4%). There were 16 queries containing redundant clauses.

Table V Processing errors in the verbal protocols for the two blocks in percentage of the total number of trials-with-errors (categories 1-6) or of the number of redundant clause queries (category 7).

Error categories	Block 1 (n=59)	Block 2 (n=21)
1 Reading a clause incorrectly	0.0%	14.3%
2 Missing a NOT while reading	49.2%	42.9%
3 Incorrect evaluation of clause	22.0%	4.8%
4 Incorrect evaluation of NOT clause	39.0%	33.3%
5 Incorrectly omitting a clause	32.2%	42.9%
6 Incorrect evaluation of the query	6.8%	4.8%
	(n=192)	(n=192)
7 Redundant evaluation of a clause	27.6%	7.8%

The data show that the percentage of errors related to a NOT is relatively high and did not decrease in the second block. The errors related to the redundant evaluation of clauses decreased in the second block. On the other hand, errors associated with incorrectly omitting a clause did not decrease.

Two further details of the processing errors are worth noting. The first concerns the effect of brackets on the processing of clauses. The presence of brackets in a query implies that the clauses within the brackets are to be grouped. Two processing errors are related to this at the query level: the omitting of a clause and the redundant evaluation of a clause (categories 5 and 7). The data show that only the omitting error is affected by the use of brackets; the error rate was 18.3% and 4.2% for mixed and mixed() query category respectively (n=120). This effect was not found for the redundant processing, the means being 23.3% and 25.8% for mixed and mixed() respectively (n=120).

The second point concerns the interaction of NOT with the clause comparator. In the processing of a NOT clause either the comparator (=,<>,<,>) has to be converted to its opposite sense before clause evaluation; or else the value (true or false) of the clause is converted after clause evaluation. The protocols indicate that, in general, the subjects convert the comparator. Conversion of "=" is simpler than conversion of "<" or ">". A common error was the conversion of "smaller than" to "greater than" instead of to "greater than or equal". The number of errors was 11.7 % (n=240) and 4.5% (n=576) for, respectively, "NOT <" or

"NOT >" and "NOT =". Logically equivalent notations "NOT=" and "< >" differed in number of errors, respectively, 4.5% (n=576) versus 1.3% (n=312).

3.3.2 **User strategies**

In general subjects processed the queries from left to right even when brackets were present in the query. Only 1.7% (n=720) of the queries were approached in a different order. Typically, protocols revealed the sequence of clause evaluation, but did not reveal directly the understanding of the logic in the query. They had the form "evaluate A, evaluate B, (evaluate C), state conclusion". There are some protocols, however, that show in detail the mechanisms of the logical operators. Table VI shows samples of two protocols.

Table VI A sample stimulus with a typical protocol (A) and a protocol that reveals the mechanisms of the logical operators (B).

	not year = 1979 and (genre=jazz or artist < > Jody Sylvian) Will Bazar classical 1965 1
(A)	"not year is 1979... correct... and the genre is jazz... not correct artist is not JS, that one is correct..." (subject mq, 2nd block, answer is correct)
(B)	"in this case we begin with an 'and' and then an OR between brackets.. the year must not be 1979 ...that is satisfied because it is 1965.. look to the other side of the 'and'... we can choose.. either the genre has to be jazz... it's not... or the artist has to be unequal to JS but that is true, thus both sides of the 'and' are satisfied by the conditions and the answer should be yes". (subject ea, 2nd block, answer is correct)

The verbal protocol A shows a fairly short protocol in which almost no references are made to logical operators or the organisation of the processing. In verbal protocol B explicit reference was made to the mechanisms of the operators and the organisation of the query. The protocol type A was the typical protocol.

In summary, the verbal protocols showed that, in general, queries were processed from left to right. Brackets did not change the order of processing. A large proportion of the errors were due to reading errors, most notably in relation to NOT clauses. Also the flipping of the greater or smaller than sign was a major source of errors. The unequal comparator ("< >") caused less problems that the logical equivalent "NOT=". The total number of errors in the second verbal protocol block was less than half that in the first block.

4 DISCUSSION

The latency and error data converge to show that subjects have increasing difficulty with longer, more complex queries: queries involving mixed (AND OR) and NOT operators were the most difficult to interpret. The inclusion of brackets in the query had a strong effect on latencies and errors. An increase in the number of logical operators from one to two had no effect on the number of errors. No differences between simple processing of AND and OR were found, in contrast with the experimental findings of Greene et al. (1990).
To interpret these findings we consider first the verbal protocol data and then develop a model aimed at identifying latency and error components in the query processing.

The verbal protocols gave insight into certain aspects of the query processing. Most importantly, they revealed how the subjects organised their analysis of the query. The verbal protocols showed that queries were always processed from left to right, including those cases in which it would have been more efficient to start at the end of the query. For instance, the query

A AND B OR C

can be analysed by identifying the part of the query that has the greatest impact on the truthness or falseness of the whole query. Here, an efficient strategy would be to evaluate C first, because if C is true then the whole query is true and the answer for the query can immediately be given. This hierarchical approach was, however, not used by the subjects.

A model of query processing can be formulated in which the basic processes are *reading* the elements of the query (the clauses and the logical operators); *comparing* clauses against the instance; *storing* the resulting values of the clauses; and *activating* the rule of the logical operator by retrieving it from memory and storing for immediate use. In addition to making provision for application of the logical operator to the truthness or falseness of the combined clauses, the model must also dictate the flow of processing. For instance, in A AND B OR C, clause B can be skipped if A is false. Thus, two other processes are: *deciding* how to go on and *computing* the value of the logical combination of the clause values.

In Fig. 3 this model of query processing is presented for a three clause query. The flow is from left to right. The first step is the processing of clause A, comprised of the basic processes of *read and compare* against the instance. The resulting TRUE/FALSE value of clause A is passed on to the *decide* block, which also requires the rule for the logical operator following the clause. The rule is obtained from the block below, *activate* log.op. The decision on how to continue depends on the combination of logical rule and value of clause A. For instance, if the clause is true and the logical operator is AND, then clause B must be processed; if the clause is false, B can be skipped.

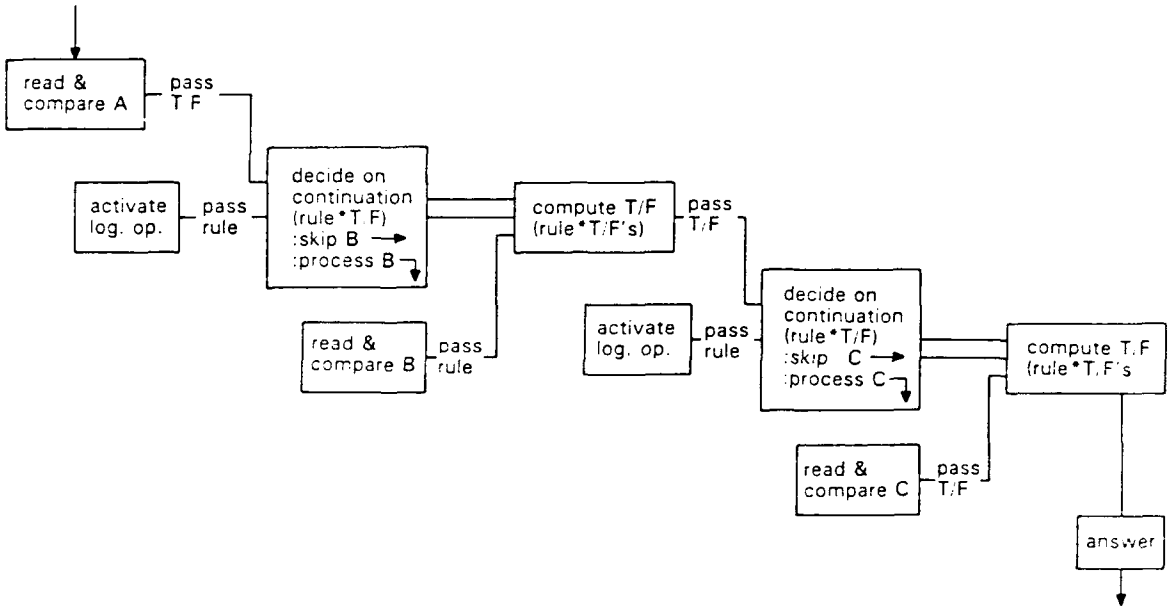


Fig. 3 Model of query processing of a three clause query. (A, B, C: the clauses; log.op.: logical operator; T/F: true/false).

In either case, *compute* produces a temporary value based on the combination of logical rule and values passed from processing of clause A and B (if done). If B is skipped, then the value of clause A alone is passed as the value of the query so far. At this stage, the next logical operator must be *activated* to *decide* how to continue. Here the same thing happens as at the former decision point. If clause C can be skipped, *compute* returns the value of the query based on clauses A and B and the answer can be given.

Given that the processing is from left to right, it is possible to specify which elements in the query must be successively processed to arrive at a conclusion on the query. In Table VII, these clauses and operators are specified from left to right for all the possible TRUE/FALSE combinations. Note that no precedence rules were used in this experiment. We assumed that subjects with no programming knowledge or specific training on logic would also lack knowledge of specific rules. The precedence rules would add an additional degree of complexity to the query processing. In fact, only the query A OR B AND C with clause A being TRUE might have been problematic, because, using the precedence rule, it has to be processed differently.

Table VII The clauses (A, B, C) that have to be processed to determine the value of the query for the different values of the clauses (T=true, F=false) given a left to right processing of the query. The logical operator to be processed is indicated by º. (NB, no precedence rules were used in the experiment).

Query categories																				
Double							Mixed								Mixed()					
AND AND			OR OR				AND OR			OR AND					AND (OR)			OR (AND)		
Clause																				
A	B	C	A	B	C		A	B	C	A	B	C			A	B	C	A	B	C
T º	T º	T/F	F º	F º	T/F		T º	F º	T/F	F º	T º	T/F			T º	F º	T/F	F º	T º	T/F
T º	F º		F º	T º			T º	T º		F º	F º				T º	T º		F º	F º	
F º	º		T º	º			F º	º	T/F	T º	º	T/F			F º			T º		

From Table VII it can be seen that with some combinations of logical operators and TRUE/FALSE values in the mixed category, more clauses need to be processed than in the other categories. In most cases, both logical operators have to be processed to arrive at a conclusion about the value of the query. In some cases in the mixed() condition, however, the value of the query can be computed after processing only the first operator.

From the processing specifications in Table VII it can be concluded that the average amount of processing will be different for the different query categories. Now, if we assume that the two main processing components are the processing of clauses and the logical analysis (activate, decide, compute) and that these two components cost the same amount of time, we can use the number of processing components as a rough estimate of the latencies. Totalling the processes, the ratio of amount of processing in the three conditions is, on average, 1 : 1.08 : 0.92 for double : mixed : mixed(). Other time aspects, like e.g. start and answer, are considered to be negligibly small compared to the two major time components. Thus, on the basis of the model and processing characteristics, the latencies in the mixed and mixed() categories can be predicted using the double category as norm. The predicted and the observed latencies are presented in Table VIII. Each level of NOT was considered as a separate case, because we do not see yet how that factor can be incorporated in the model. The predictions were confined to queries with equal number of operators (three). Extrapolation to the single operator queries showed underestimation of processing time.

Table VIII Predicted and (between brackets) observed average response latencies (s) with the Double data as basis.

	Query categories	
Double	Mixed	Mixed()
0-NOT (8.3)	8.9 (8.9)	7.6 (8.3)
1-NOT (9.4)	10.2 (10.3)	8.6 (8.9)
2-NOT (9.9)	10.7 (10.7)	9.1 (9.1)

This analysis shows that the effect of brackets is to reduce the amount of processing needed in the mixed() category, since processing of the second operator can sometimes be omitted. It is not clear precisely what further effect the brackets might have on processing; further experiments investigating this aspect should control for the number of clauses processed in the bracket and non-bracket cases.

These comparisons show that latencies can be predicted with some accuracy. The intention in using this model is to show that one way to explain the latency results is on the basis of the number of processing components in the query. Further research is needed to investigate the assumption that the time for the different processes are equal.

How can the model help to interpret the error data? From the protocol data we saw that the number of reading errors was relatively high in proportion to other errors. Reading errors occur during the processing of the clauses, in particular in combination with the NOT operator. Following the same line of reasoning as with the latency data, the number of clauses to be processed (from Table VII) could be used to predict the error data in the speed trials. The ratio of the number of clauses processed in the three conditions is thus computed as 1 : 1.33 : 1. However, the error rate predicted using these assumptions differs from the observed data. The fact that the double and single conditions did not differ in number of errors (despite a two-fold difference in number of clauses processed) also argues against using amount of processing as a basis for error prediction (see Fig. 2). A significant difference between double and mixed queries lies in the fact that the same logical rule is invoked in the double case whereas two different rules are invoked in the mixed case. Interference between the two different rules might explain the rise of errors in the mixed condition. The brackets in mixed() might have helped the subject to mentally segregate the processing of the two operators, resulting in a large reduction of errors. It is unclear which factor plays a role in the interpretation of queries containing brackets.

A somewhat surprising result concerned differences between university and high school students. Subjects were selected as having no specific logical training or programming experience. No a priori differences were expected between the university and high school students. Indeed, both groups started off at the same processing speed and did not differ in the error data. Both groups improved performance in term of errors. However, the university students became relatively faster in the successive blocks. In terms of speed-accuracy tradeoff their performance became more efficient than that of the high school students. Both groups learned during the experiment although there was no feedback given during the experiment.

A remark is necessary concerning the methodology. The combination of latency and error measurement and verbal protocols was a good methodology for addressing the problem of the interpretation logical operators. The protocols gave insight into details of the query processing. However, the application of the logical rules was not explicitly referred to in the protocols. One plausible explanation for this is that the problem the subjects had to solve did not evoke deliberate problem solving, because the rules were too simple and were therefore processed quickly. When processes are fast, fewer verbal references are made to those processes (Ericsson & Simon, 1984).

Most studies have addressed query production, i.e. the creation of a query. In our study we studied a process that is assumed to be part of query production, namely, query interpretation. The reason for selection of this strategy was that the understanding of logical operators should be separated from other factors in querying production, like knowledge of the dataset or understanding of the data model. We regard query interpretation as a prerequisite for query production. Understanding logical operators is necessary for the active use of them. The difference between these two task situations is indicative of the extra processes that apply in query formulation investigated in other studies. Our results indicate that people can successfully interpret logical operators. However, the subjects did not make use of efficient strategies to solve the problems. The number of errors was low in this task situation compared to the results from studies in query generation. (Greene et al (1990) reported an error rate of 47.5 % in the generation of SQL-type of queries with AND OR operators.) In this study we have used relative simple problems. More complex problems could force people to explicitly organise the processing of the query. Further studies on logical operators should address that organisation process and the question of how to support it.

REFERENCES

Borgman, C.L. (1986). The user's mental model of an information retrieval system: an experiment on a prototype online catalog. International Journal of Man-Machine Studies 24, 47-64.

Broadbent, D.E. & Broadbent, M.H.P. (1978). The allocation of descriptor terms by individuals in a simulated retrieval system. Ergonomics 21(5), 343-354.

Brosey, M. & Shneiderman, B. (1978). Two experimental comparisons of relational and hierarchical database models. International Journal of Man-Machine Studies 10, 625-637.

Chamberlin, D.D., Astrahan, M.M., Eswaran, K.P., Griffiths, P.P., Lorie, R.A., Mehl, J.W., Reisner, P. & Wade, B.W. (1976). SEQUEL 2: A unified approach to data definition, manipulation and control. IBM Journal of Research and Development 20, 560-575.

Cooper, W. (1988). Getting beyond Boole. Information Processing & Management 24(3), 243-248.

Ericsson, K.A. & Simon, H.A. (1984). Protocol Analysis: Verbal Reports as Data. Cambridge, MA: MIT Press.

Greene, S.L., Devlin, S.J., Cannata, P.E. & Gomez, L.M. (1990). No IFs, ANDs, or ORs: A study of database querying. International Journal of Man-Machine Studies 32, 303-326.

Jarke, M. & Vassiliou, Y. (1985). A framework for choosing a database query language. Computing Surveys 17(3), 313-340.

Katzeff, C. (1986). Dealing with a database query language in a new situation. International Journal of Man-Machine Studies 25, 1-17.

Lochovsky, F.H. & Tsichritzis, D.C. (1977). User performance considerations in DBMS selection. Proceedings of ACM SIGMOD (pp. 124-134). New York: Association for Computing Machinery.

Marchionini, G. (1989). Making the transition from print to electronic encyclopaedias: adaptation of mental models. International Journal of Man-Machine Studies 30, 591-618.

Michard, A. (1982). Graphical presentation of boolean expressions in a database query language: design notes and an ergonomic evaluation. Behaviour and Information Technology 1(3), 279-288.

Ogden, W. & Kaplan, C. (1986). The use of "and" and "or" in a natural language computer interface. Proceedings of the Human Factors Society 30th Annual Meeting (pp. 829-833). Santa Monica, CA: Human Factors Society.

Reisner, P., Boyce, R.F. & Chamberlin, D.D. (1975). Human factors evaluation of two data base query languages - Square and Sequel. Proceedings of the National Computer Conference (pp. 447-452). Arlington: AFIPS Press.

Reisner, P. (1977). Use of psychological experimentation as an aid to development of a query language. IEEE Transactions on Software Engineering SE-3(3), 218-229.

Thomas, J.C. & Gould, J.D. (1975). A psychological study of query by example. Proceedings of the National Computer Conference (pp. 449-445). Arlington: AFIPS Press.

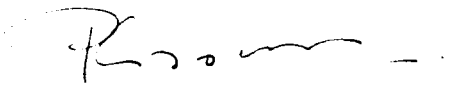
Thompson, R.H. & Croft, W.B. (1989). Support for browsing in an intelligent text retrieval system. International Journal of Man-Machine Studies 30, 639-668.

Vassiliou, Y. & Jarke, M. (1984). Query languages - A taxonomy. In Y. Vassiliou (Ed.), Human Factors and Interactive Computer Systems (pp. 47-82). Norwood, NJ: Ablex.

Wason, P.C. & Johnson-Laird, P.N. (1972). Psychology of Reasoning: Structure and Content. Cambridge, MA: Harvard University Press.

Welty, C. (1985). Correcting user errors in SQL. International Journal of Man-Machine Studies 22, 463-477.

Soesterberg, March 22, 1991



Drs. P.J.M.D. Essens

APPENDIX Queries in the verbal protocol blocks (the first block equals the second block).

	Query categories	
	Single	Double
0-NOT	= and > < > or < >	= and > and = = or = or <
1-NOT	not = and > = and not = not = or = < or not =	not < and = and = < > and not < and = = and < > and not = not < > or = or = = or not < or < > < or = or not >
2-NOT	not = and not = not = or not =	not = and not = and = not = and < > and not > not = or not > or = = or not > or not <

	Query categories	
	Mixed	Mixed()
0-NOT	= and < > or > > or = and >	= and (= or =) < > or (= and < >)
1-NOT	not = and = or < > < > and not = or > < > or not = and = = or < > and not >	not = and (= or < >) < and (= or not =) not > or (= and =) = or (= and not =)
2-NOT	not = and not = or = = and not = or not = not > or not = and < > not < or = and not >	not = and (not = or >) > and (not = or not =) not = or (not = and =) = or (not = and not =)

REPORT DOCUMENTATION PAGE

1. DEFENCE REPORT NUMBER (MOD-NL)	2. RECIPIENT'S ACCESSION NUMBER	3. PERFORMING ORGANIZATION REPORT NUMBER
TD 91-0043		IZF 1991 B-2
4. PROJECT/TASK/WORK UNIT NO.	**5. CONTRACT NUMBER**	**6. REPORT DATE**
733.1	B91-34	March 22, 1991
7. NUMBER OF PAGES	**8. NUMBER OF REFERENCES**	**9. TYPE OF REPORT AND DATES COVERED**
29	20	Final

10. TITLE AND SUBTITLE

An exploratory study of the interpretation of logical operators in database querying

11. AUTHOR(S)

P.J.M.D. Essens, C.A. McCann and M.A. Hartevelt

12. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

TNO Institute for Perception
Kampweg 5
3769 DE SOESTERBERG

13. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

TNO Defence Research
Schoemakerstraat 97
2628 VK Delft

14. SUPPLEMENTARY NOTES

15. ABSTRACT (MAXIMUM 200 WORDS, 1044 BYTE)

The use of logical operators in query languages is considered to be a major source of user problems in database querying. The present study investigated whether people untrained in logic could successfully interpret logical operators; and, how errors and latencies are related to the structure of the query. In an experiment, the logical complexity of an SQL-style query formulation was varied in using AND, OR, and NOT operators in either single or combined form. The latency and error data converged to show that subjects had increasing difficulty with queries constructed with a combination of different operators. The inclusion of brackets had a strong positive effect on task performance. Verbal protocols were used to identify sources of errors in query processing. A model of query processing was formulated and predictions latencies and errors on the basis of processing components were tested.

16. DESCRIPTORS	IDENTIFIERS
Human Factors Engineering	Database Querying
Man-Machine Interactions	Boolean Operators
Human Errors	Query Complexity
	Cognitive Skills

17a. SECURITY CLASSIFICATION (OF REPORT)	17b. SECURITY CLASSIFICATION (OF PAGE)	17c. SECURITY CLASSIFICATION (OF ABSTRACT)
-	-	-
18. DISTRIBUTION/AVAILABILITY STATEMENT		17d. SECURITY CLASSIFICATION (OF TITLES)
Unlimited availability		-

VERZENDLIJST

1. Hoofddirecteur van TNO-Defensieonderzoek

2. Directie Wetenschappelijk Onderzoek en Ontwikkeling Defensie

3. { Hoofd Wetenschappelijk Onderzoek KL
   Plv. Hoofd Wetenschappelijk Onderzoek KL

4, 5. Hoofd Wetenschappelijk Onderzoek KLu

6. { Hoofd Wetenschappelijk Onderzoek KM
   Plv. Hoofd Wetenschappelijk Onderzoek KM

7, 8, 9. Hoofd van het Wetensch. en Techn. Doc.- en Inform. Centrum voor de Krijgsmacht

10. Dr. D.G. Pearce, Defence and Civil Institute of Environmental Medicine, North York, Ontario, Canada

---

Extra exemplaren van dit rapport kunnen worden aangevraagd door tussenkomst van de HWOs of de DWOO.
